REMARKS

The application includes claims 1-23 prior to entering this amendment.

The examiner objects to claim 9 for informalities. Claim 24 has been added. Claims 1-24 remain in the case for reconsideration. Reconsideration is requested.

Claim Objections

Method claim 9 has been amended to depend upon method claim 5 and therefore overcomes the objection.

Claim Rejections - 35 U.S.C. § 103

The examiner rejects claims 1-2, 4-6, and 8-23 under 35 U.S.C. § 103(a) as being unpatentable over Walker in view of Bhattacharjee et al., and further in view of Ng (U.S. Pat No 4,627,019) as applied to claims 1-2, 4-6, and 8-23 above.

The examiner acknowledges that Walker and Bhattacharjee do not disclose multiple different activities with a same transaction, each of the activities consisting of an associated subgroup of program instructions for the same transaction. However, the examiner states that this is disclosed in Ng at col. 5, lines 18-48.

The rejection is respectfully traversed. However, claims 1, 5, and 23 have been amended to further distinguish over Ng.

Amended claim 1 recites:

a processor associating multiple different activities with a same transaction, each of the multiple different activities consisting of a separate different associated subgroup of program instructions for the same transaction, each different subgroup of program instructions initiating a different associated subgroup of read and/or write actions on an associated group of data.

Ng has nothing to do with associating multiple different activities with a same transaction, where each of the multiple different activities consists of a separate different associated subgroup of program instructions for the same transaction and each different subgroup of program instructions initiates a different associated subgroup of read and/or write actions on an associated group of data as recited in claim 1.

Ng does discuss multiple different database transactions that may be contemporaneously active. However, there is no suggestion that multiple different transactions are ever associated together with each other. The transactions in Ng are only associated with a particular data relation pointer 210 (FIG. 5) and have no concept of multiple different subgroups of read and/or write operations all associated with the same transaction as recited in claim 1.

In fact, the transactions in Ng cannot be associated with multiple different activities as recited in claim 1. Refer to col. 7, lines 16-34 of Ng where each transaction block 110 in FIG. 4 of Ng only has a single transaction Identifier 113 that identifies a particular type of transaction, such as a read only transaction or an update transaction (col. 7, lines 17-20). There is no suggestion in Ng that multiple different activities, each having different subgroups of read/write operations, are associated for the same transaction identifier as recited in claim 1. If fact, that would be impossible since each transaction identifier 110 in Ng by definition defines a particular transaction operation.

Conversely, the transaction as recited in claim 1 is a generic number used by *a processor to associate multiple different activities with a same transaction, each of the multiple different activities consisting of a separate different associated subgroup of program instructions for the same transaction, each different subgroup of program instructions initiating a different associated subgroup of read and/or write actions on an associated group of data.* The transaction as recited in claim 1 is used to associate different subgroups of database activities together, not to identify a particular transaction operation as defined in Ng.

Further, Ng only allows a single read transaction or update transaction and a single relation block per transaction (col. 6, lines 55-61). Thus, the transactions and relation blocks in Ng does not teach multiple different activities associated with a same transaction where each of the multiple different activities consists of a separate different associated subgroup of read and/or write program instructions for the same transaction as recited in claim 1.

Claim 1 also recites: *the processor associating separate lock durations with each different subgroup of program instructions associated with each of the different activities associated with the same transaction, and maintaining the multiple different locks for all of the subgroup of program instructions associated with the same activities and then releasing all of the multiple different locks associated with the same activities together only when all of the*

subgroup of program instructions associated with the same activities are completed so that all of the multiple different locks associated with the same activities have a same lock duration.

These elements are clearly shown in FIGS. 2-7 and explained in the specification at page 6, lines 23-25: "The LM 3 releases all of the locks for the given activity 14 together. . . . Thus all locks associated with activity 14 are of this "activity duration". The specification at page 7, lines 20 also explains: "Again it is faster to unlock all tuples in a prefetch group in one operation rather than unlocking the tuples one by one. This is ideally satisfied by the notion of activities."

This is contrary to Ng and other conventional lock managers that releases locks on data items serially whenever the program instructions associated with those data items are completed.

Ng does not even discuss locks, much less *associating separate lock durations with each different subgroup of program instructions associated with each of the different activities associated with the same transaction, and maintaining the multiple different locks for all of the subgroup of program instructions associated with the same activities and then releasing all of the multiple different locks associated with the same activities together only when all of the subgroup of program instructions associated with the same activities are completed* as recited in claim 1.

Ng discusses different dictionaries that include pointers that are used to point to different data versions (see FIGS. 5 and 6 of Ng). Nowhere does Ng discuss a programming scheme for associating different subgroups of read and/or wire actions with a same activity and then also associating the multiple different activities with a same transaction as recited in the first element of claim 1.

In Ng, the relation IDs 210-1 and 210-2 in FIG. 5 are merely pointers that determine access to a current version of a data relation 500-1 (FIG. 6). A relation is defined in Ng as follows: "The data are stored as relations and the storage is implemented in a two-level tree-like structure. The data comprising one relation are stored in data blocks . . of memory 21 and a head block 500-2 for that relation contains index entries . . each defining the location of one of the data blocks . . " (col. 4, lines 53-63).

Neither the transactions in Ng or the dictionaries 100, 200, 300, and 400 in FIG. 4 of Ng maintain separate lock durations for different subgroups of program instructions associated with the same activities as recited in claim 1. See Ng at col. 7, lines 32-34 where at most one update transaction is allowed to become active to access a give relation at any given time.

In Ng, an access block in access dictionary 400 and a relation block 210 in FIG. 4 is allocated for a transaction, but the pointers in the dictionaries 400 and 200 are not multiple different associated subgroups of read and/or write program instructions that are each associated with multiple different locks for all of the subgroup of program instructions associated with the same activities.

Contrary to the complex pointer scheme described in Ng, claim 1 associates multiple different subgroups of program instructions with the same activities. This allows all of the multiple different locks associated with the same activities to be released together when all of the different subgroup of program instructions associated with that activity are completed.

The amended elements in claim 1 are also not suggested in Walker or Bhattacharjee. Accordingly, claim 1 is allowable under 35 U.S.C. § 103(a) over Walker in view of Bhattacharjee et al., and further in view of Ng.

The other independent claims 10, 14, and 18 include at least some similar element as claim 1 and therefore are allowable for at least some of the same reasons as claim 1. For example, claim 10 recites the processor assigning multiple locks to the multiple data items corresponding with the operations performed on the multiple data items and only releasing the multiple locks when all of the multiple operations are completed for all of the database access instructions assigned to the same activity identifiers.

Claim 4 recites releasing all of the multiple different locks associated with the same activities in one operation only when all of the multiple different subgroup of program instructions associated with the same activities are completed. This is described at page 7, line 20. A single operation that releases multiple different locks for different subgroups of program instruction is possible because all of the subgroups are assigned a same activity identifier.

No combination of Walker, Bhattacharjee, and Ng suggest any technique for releasing multiple different locks in one operation as recited in claim 4. This is contrary to a conventional way of serially releasing locks one-by-one on data items immediately when the instructions accessing those data items are completed

Claim 23 recites:

assigning a first activity identifier and a transaction identifier to a first group of database access instructions for a transaction;

assigning a first set of multiple locks to a first set of data items accessed by the first group of database access instructions, the multiple locks assigned to different ones of the first set of data items according to the first subgroup of database access instructions;

identifying a second subset of data items from the first set of data items according to the first group of database access instructions;

assigning a second activity identifier and the same transaction identifier to a second group of database access instructions for the same transaction that modify the second subset of data items identified by the first group of database access instructions;

assigning a second set of multiple locks to the second subset of data items, the second set of multiple locks having a different lock duration than the first set of multiple locks;

releasing the first set of multiple locks for all of the first set of data items that are not part of the second subset of data items only after the second group of database access instructions have completed; and

releasing the entire second set of locks only when all of the operations for the second group of database access instructions have completed modification of the second subset of data items.

This is clearly explained on pages 8 and 9 and specifically at page 9, lines 19-21 where an activity #1 ends only after the completion of activity #2 and the end of activity #1 automatically releases locks on the data items not verified by activity #2.

Neither Walker, Bhattacharjee, or Ng suggest releasing a first set of multiple locks for all of a <u>first set of data items associated with in a first activity</u> only after the instructions associated with a <u>second different activity</u> are completed.

This is counter to conventional database locking where locks associated with a first activity are normally released after the operations associated with that same first activity were completed. Not when a second different activity is completed.
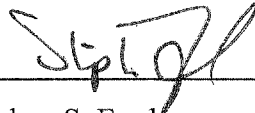
Conclusion

For the foregoing reasons, reconsideration and allowance of claims 1-24 of the application as amended is requested. The Examiner is encouraged to telephone the undersigned at (503) 224-2170 if it appears that an interview would be helpful in advancing the case.

Customer No. 73552

Respectfully submitted,

STOLOWITZ FORD COWGER LLP

_____

Stephen S. Ford

Reg. No. 35,319

STOLOWITZ FORD COWGER LLP
621 SW Morrison Street, Suite 600
Portland, OR 97205
(503) 224-2170